Patterns

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
5.2.47
Information Systems
March 9, 2008

Goal

Refactoring
- What is refactoring
- Arguments for refactoring
- Bad Smells
- Obstacles

Tutorial on MDD and UP

Design Patterns
- What is Pattern
- Types of Patterns
- Benefits of Patterns

What is a Pattern?

A pattern addresses a recurring problem that arises in specific situations.

Patterns document existing, well-proven design experience.

Patterns identify and specify abstractions that are above the level of single classes and instances.

What is a Pattern ?

Patterns provide a common vocabulary and understanding for design principles.
Patterns are a means of documenting software architectures.
Patterns support the construction of software with defined properties.
Patterns help you build complex and heterogeneous software architectures.
Patterns help you manage software complexity.

Types of Patterns

Design Patterns
Analysis patterns
Software architecture patterns
Process patterns
Organizational patterns
...

Design Patterns

Design patterns provide **abstract, reusable "micro-architectures"** that can be applied ("instantiated") to resolve specific design issues (forces) in previously-used, high-quality ways

GoF (Gang of Four – Gamma, Helm, Johnson, Vlissides) defined widely used 23 design patterns like Factory, Observer, Visitor, State, Strategy etc.

Patterns Descriptions

Name and intent
Problem and context
Forces addressed
Abstract description of structure and collaborations in solution
Positive and negative consequences of use
Implementation guidelines and sample code
Known uses and related patterns

Analysis Patterns

By Martin Fowler :

Patterns that reflect **conceptual structures** of business processes rather than actual software implementations.

Fowler uses a simple, specialized notation (very similar to entity-relationship diagram notation) to depict graphical models of analysis patterns.

Software Architecture Patterns

Architectural patterns are templates for **concrete** software architectures. They specify the system-wide structural properties of an application, and have an impact on the architecture of its subsystems.

Other Pattern Types

Process Patterns

- patterns that deal with software development **process issues**
- work by Scott Ambler

Organizational Patterns

- patterns that deal with **organizational issues** that arise in software development teams, groups and departments
- these can often be related to software process patterns

Potential benefits of Patterns

Provides a common **vocabulary** and **understanding** of design elements for software designers

Increases **productivity** in design process due to "design reuse"

Promotes **consistency** and high quality of system designs and architectures due to application of tested design expertise and solutions embodied by patterns

Benefits of Patterns

allows all levels of designers, from novice to expert, to gain these productivity, quality and consistency benefits

Concerns

Training and education in common and proprietary patterns

- benefits are dependent upon architects, analysts and designers understanding the patterns to be used
- benefits are enhanced significantly only if ALL software development personnel understand the patterns as a common "design vocabulary"
- such training can be costly, and in many cases is proprietary and cannot be obtained externally

Concerns

Evolution and maintenance to insure continued value

- specific funding and effort must be directed toward maintenance and evolution of patterns as reusable assets or they tend to devolve into project/application-specific artifacts with dramatically reduced reusability characteristics
- the necessary funding, technical or organizational infrastructure supports may not exist to allow effective maintenance and evolution of patterns within an organization to insure continued high benefits

AntiPatterns (from Brown, et al.)

An AntiPattern is a literary form that describes a commonly occurring solution to a problem that generates decidedly negative consequences.

Primal Forces

AntiPatterns, like other patterns, deal with forces (concerns, issues) that exist in a specific problem setting

Vertical forces are problem domain-specific

Horizontal forces are applicable across multiple domains or problem settings

Primal Forces

Primal Forces are horizontal forces that are pervasive in software architecture and development. They include:

- Management of functionality: meeting the requirements
- Management of performance: meeting required speed of operation
- Management of complexity: defining abstractions
- Management of change: controlling the evolution of software
- Management of IT resources: controlling the use and implementation of people and IT artifacts
- Management of technology transfer: controlling technology change

Patterns and Refactoring

Patterns can help to improve the code
Patterns emerge from refactoring
Combination of patterns can lead to refactoring